Operating System Assignment

Ritam Sengupta, CSE (5th Sem);
Roll-33200118019;
PCC CS-502

1) Operating System :

An operating system is an interface between a computer user and computer hardware which performs all the basic tasks like file management, memory management, process management, handling output and ~~output~~ input. and controlling peripheral devices such as disk drives and printers.

Functions of an OS :

An operating system is an interface between computer's hardware and user himself whose functions are;

i) Manage the computer resources such as :
- a) CPU;
- b) Memory;
- c) Disk Drives & printers

ii) Establish a user interface (UI);

iii) Execute and provide services for applications software.

2) Multitasking :

Multitasking is the process of running more than 1 program (set of instructions) in one computer at the same time. It is used to keep all of a computer's resources at work as much of the time as possible.

In terms of OS ; Allowing a user to perform more than one computer task (such as the operation of an application program) at a time.

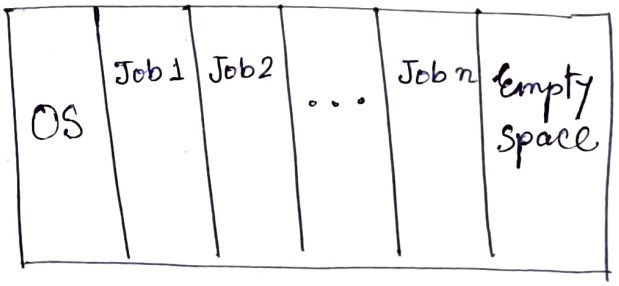
The OS is able to keep track of where you are in these tasks and go from one to other without losing information.

MS Windows 2000, IBM OS/390 are some examples of multitasking OS's.

(3) Multiprogramming:

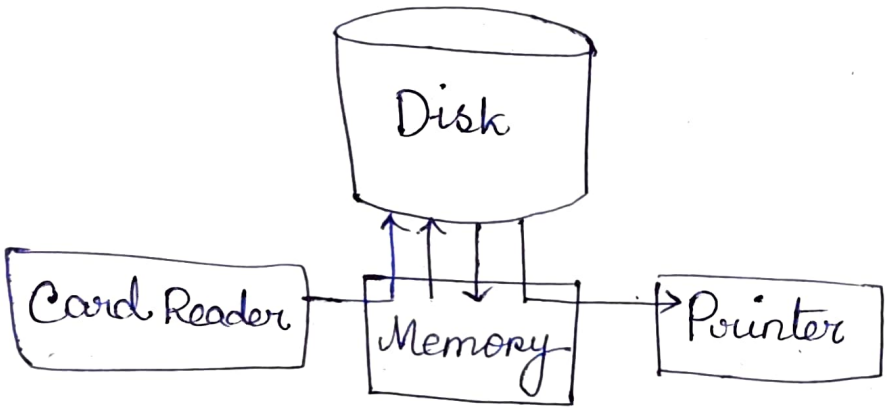
It is the ability of an OS to execute more than one task/program at a time. It is used to keep all of a computer's resources at work as much of the time as possible on a single processor machine. More than one task/program/job/process can reside into the main memory at one point of time. A computer running reside into the main memory at one point of time. A computer running excel and firefox browser simultaneously is an example of multiprogramming.

Memory layout for Multiprogramming System:



(4) SPOOLING:

Simultaneous Peripheral Operations Online refers to putting data of various I/O jobs in buffer. (It becomes possible to have the computer read data from a tape, write data to disk and to write out to a tape printer while it is doing its computing task). This buffer is a special area in memory or hard disk which is accessible to I/O drives.

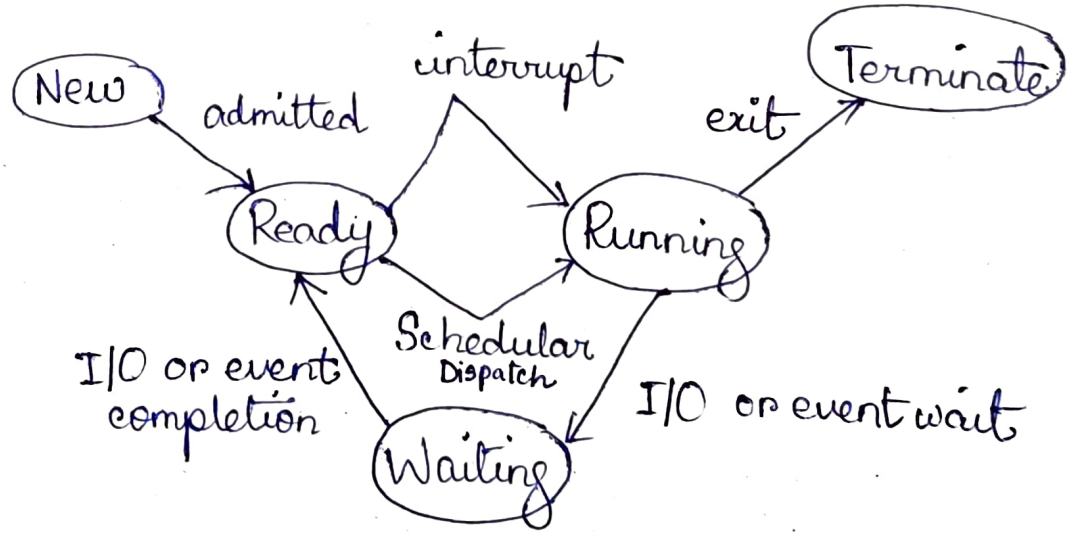


(5) Process :

A process is the instance of a computer program that is being executed by one or many threads. It contains the program code and its activity.

States :

- NEW → Process is being created
- READY → Process is waiting to be assigned to a processor.
- RUNNING → Instructions are being executed.
- WAITING → Process is waiting for some event to occur
- TERMINATED → The process has finished execution.

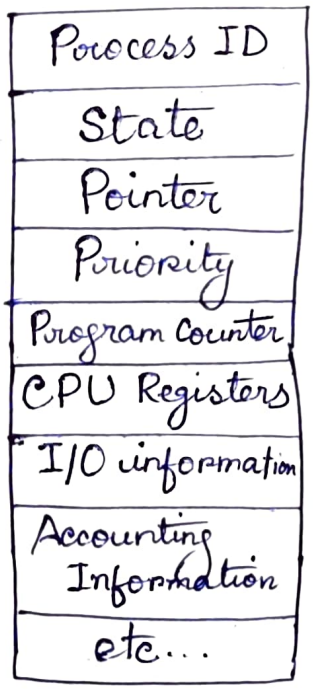


Ritam Sengupta Roll-19

(6) Process Control Block (PCB):-

It is a data structure which contains:-

- Process State: Can be running, waiting etc.
- Process ID and the parent process ID.
- CPU registers and Program Counter. Program Counter holds the address of the next instruction to be executed for that process.
- CPU scheduling information: such as priority information and pointers to scheduling queues.
- Memory Management information: i.e, page tables or segment tables.
- Accounting Information: The user and Kernel CPU time consumed, account numbers, limits, etc.
- I/O status information: Devices allocated, open file tables, etc.



(7) Difference between, long, short & medium term scheduler,

Sl no.	Long-Term Scheduler	Short-term Scheduler	Medium-Term Scheduler
1>	It is a Job Scheduler.	It is a CPU scheduler.	It is a process swapping scheduler.
2>	It takes process from the job pool.	It takes the process from the ready state.	It takes process from running or wait/dead state.
3>	Speed is lesser than short-term.	It is the fastest among two others.	Speed is in between the other two.
4>	Controls the degree of multiprogramming.	Has less control over the degree of multiprogramming	Reduces the degree of multiprogramming

(8) Context Switching :

Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. The task is known as Context Switch.

Example : In the Linux Kernel, context switching involves switching registers, stack pointer, program counter, flushing the translation lookaside buffer (TLB) and loading the page table of the next process to run (unless the old process shares the memory with the new)

(9) Thread :- A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers

which hold its current working variables, and a stack which contains the execution history.

- Process Vs Thread :

Process	Thread
1) Process is heavy-weight or resource intensive.	1) It is a light weight, taking lesser resources than a process.
2) Process switching needs interaction with operating system.	2) Thread switching does not need to interact with operating system.
3) In multiple processing environments, each process executes the same code but has its own memory and file resources.	3) All threads can share same set of open files, child processes.

- Benefits of Thread :
  1) minimize the context switching.
  2) provides concurrency within a process.
  3) Efficient communication.
  4) Allows utilization of multiprocessor architectures to a greater scale and efficiency.

(10) User Level Thread : The thread management is not aware of the existence of threads. The thread library contains code for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts.

Ritam Sengupta — Roll-19

- **Kernel Level Thread:** Thread management is done by the kernel. There is no thread management code in application area. Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

- **Difference;**

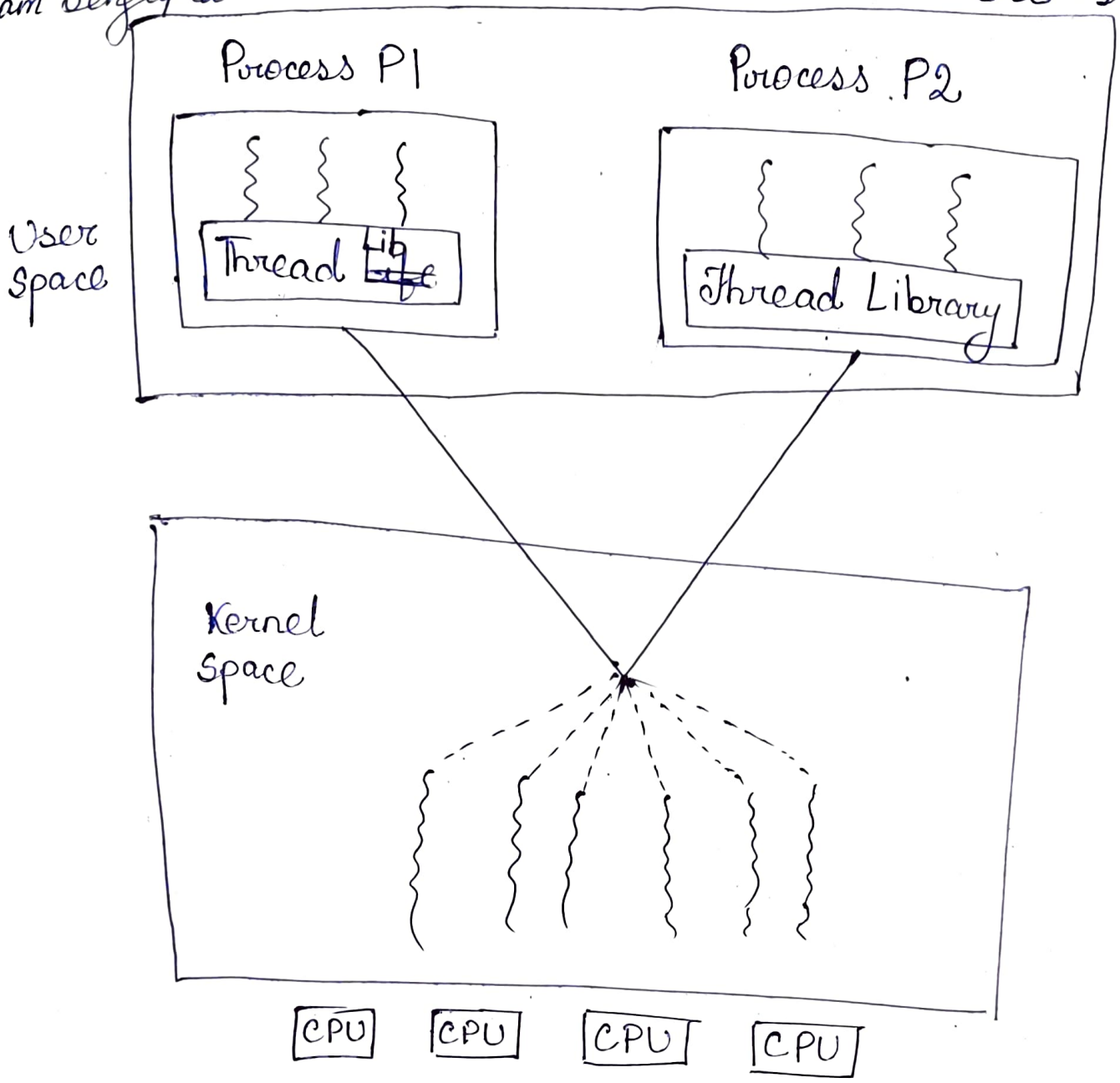
User Level Threads	Kernel Level Thread
1) Are faster to create and manage.	1) Threads are slower to create & manage.
2) Implementation is by a thread library at the user level.	2) OS supports creation of Kernel threads.
3) It's generic and can run on any OS.	3) It is specific to the OS.
4) Multi-threaded applications cannot take advantage of multiprocessing.	4) Kernel routines themselves can be multithreaded

(11). i) **Many to many Model;**

Ans→ The many to many model multiplexes any number of user threads onto an equal or smaller number of kernel threads

Many to many Model of 6 user level threads multiplexing with 6 kernel level threads.

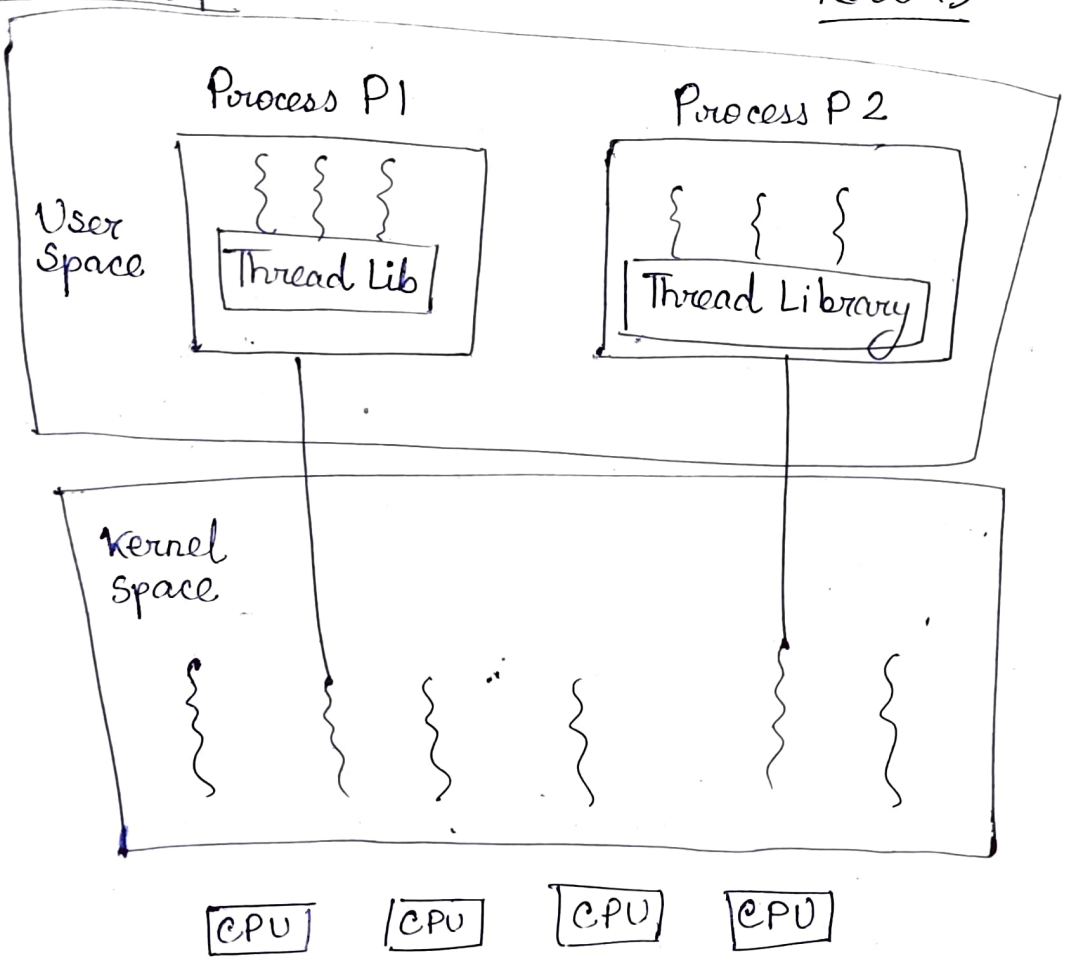
Ritam Sengupta



Many to One Model

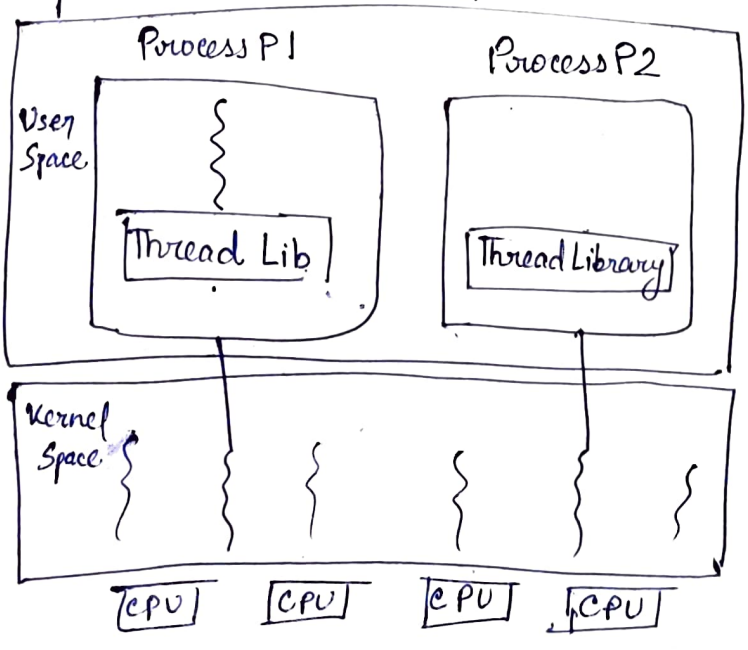
Many-to-one model maps many user level threads to one kernel-level thread. Thread management is done in user space by the thread library. When thread makes a blocking system call, the entire process will be blocked only one thread can access the kernel at a time, so multiple threads are unable to run in a parallel on multiprocessors.

Ritam Sengupta



One to one Model :

There is one-to-one relationship of user level thread to the kernel-level thread. This model provides more concurrency than the many to one model. It also allows another thread to run when a thread makes a blocking system call. It supports multiple threads to execute in parallel on microprocessors.



Ritam Sengupta

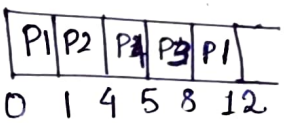
Roll-19

(12)

Process	Arrival Time	Burst Time
P1	0	5
P2	1	3
P3	2	3
P4	4	1

Process	AT	BT	CT	TAT	WT	RT
P1	0	5	12	12	7	0
P2	1	3	4	3	0	0
P3	2	3	8	6	3	8
P4	4	1	5	1	0	0

Gnatt Chart



Avg TAT = (12+3+6+1)/4
= 5.5

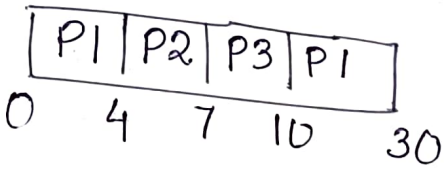
Avg Waiting Time = (7+0+3+0)/4
= 2.5

Avg Response Time = (0+0+3+0)/4
= 0.75

(13)

Process	Burst Time
P1	24
P2	3
P3	3

Gnatt Chart :



Process	BT	CT	WT
P1	24	30	6
P2	3	7	4
P3	7	10	7

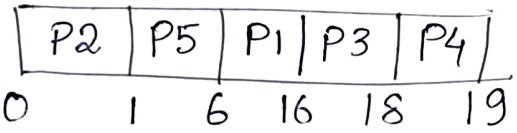
Avg waiting Time = 17/3
= 5.67

Ritam Sengupta

(14)

Process	Burst time	Priority
P1	10	3
P2	1	1
P3	2	3
P4	1	4
P5	5	2

Ans → Gnatt Chart,



Avg waiting time = (1+6+16+18)/5
+0

= 8.2

(15) Semaphore:

A semaphore is an integer variable, that is used to solve the critical section problem by using two atomic operations wait() and signal().

Wait()	Signal()
1) This operation is used to acquire.	1) This operation is used to release the lock

wait(S){
while (S<=0);
S--;
}
	signal(S){
	++S;
	}

Ritam Sengupta Roll-19

· Solution of Readers / Writers Problem :

```
Shared data structures ;
Semaphore rw_mutex=1 ;
semaphore mutex = 1 ;
int read_count=0;
```

Writer Code :

```
while (true){
 wait (rw_mutex);
 /*writing is performed*/
  signal (rw_mutex);
}
```

Reader Code :

```
while (true){
 wait (mutex);
 read_count++;
 if (read_count==1)
    wait (rw_mutex);
 signal (mutex);
/*reading is performed*/
 wait (mutex);
 read_count--;
 if (read_count==0)
    signal (rw_mutex);
 signal (mutex);
}
```

Ritam Sengupta

Roll-19

(16) Dining Philosophers problem using semaphore:

```
semaphore chopstick[5];
while(true){
    wait(chopstick[i]);
    wait(chopstick[(i+1)%5]);
    /* eat for a while */
    signal(chopstick[i]);
    signal(chopstick[(i+1)%5]);
```

To avoid deadlock, do either one of the following;

1) Allow at most four philosophers;
2) Allow a particular to pick up both chopsticks or none.
3) Use asymmetric solution odd numbered philosopher picks left chopstick then right and even numbered philosopher picks right chopstick, then left.

(17) Producer-Consumer Problem using Semaphore:

```
Semaphore empty = 1;
Semaphore full = 1;
Semaphore mutex = 1;
Producer
    while(true){
        wait(empty);
        wait(~~empty~~ mutex);
    /* Producer an item to buffer*/
        signal(mutex);
        signal(full);
    }
```

Ritam Sengupta

Roll-19

Consumer:

```
while (true){
    wait (full);
    wait (mutex);
/* Consume an item from buffer*/
    signal (mutex);
    signal (empty);
}
```